Detecting anomalies in silicon wafer manufactoring

Gabriel Scozzarro

25/05/2021

1.0 Introduction

In electronics, a wafer (also called a slice or substrate)[1] is a thin slice of semiconductor, such as a crystalline silicon (c-Si), used for the fabrication of integrated circuits and, in photovoltaics, to manufacture solar cells. The wafer serves as the substrate for microelectronic devices built in and upon the wafer. It undergoes many microfabrication processes, such as doping, ion implantation, etching, thin-film deposition of various materials, and photolithographic patterning. Finally, the individual microcircuits are separated by wafer dicing and packaged as an integrated circuit.

Silicon wafer manufacturing facilities are equipped with many sensors which monitor the manufacturing process and the semiconductors that are made down the production line. Currently, much of the data that is generated by the sensors is only used for troubleshooting when a problem arises. A single manufacturing process has thousands, of different parameters from sensors, so efficiently determining the source of a problem in a specific process is very difficult.

It's also necessary to specify that in a such complex manufacturing line a failure rate is physiological and inevitable but if you can predict the failure of a specific production lot starting from the sensors readings, it will be possible to ensure a reliable delivery to your clients and better schedule the time to delivery and the entire production line,

1.1 Scope of work & Business task

Detecting anomalies during all those manufacturing processes is crucial. However, current methods of anomaly detection often rely on simple excursion detection methods, and manual inspection of machine sensor data to determine the cause of a problem. In order to improve semiconductor production line quality, machine learning tools can be developed for more thorough and accurate anomaly detection.

2.0 Data

The data set provided is avaible on **Kaggle** it contains all the sensors reading (~1500) during the production of 1 silicon wafer and a 0-1 logic feature stating the detection of anomalies in that specific wafer.

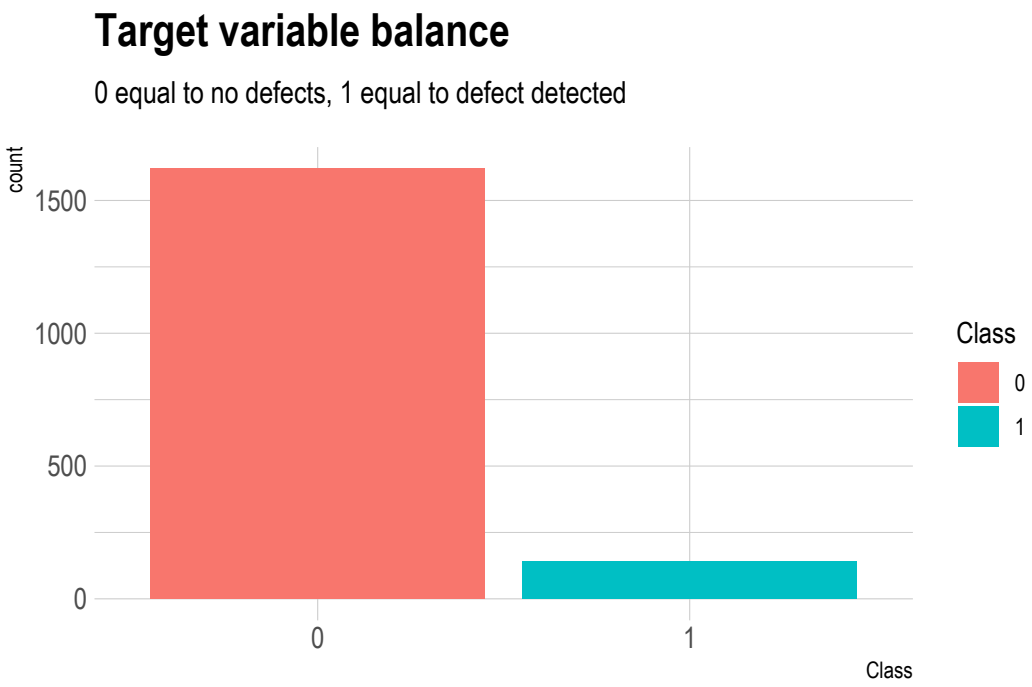
3.0 Tools and process

The analysis was performed using R coding language. A complete list of R packages, a data log and the source code are all avaible on the project github repository at this **link**.

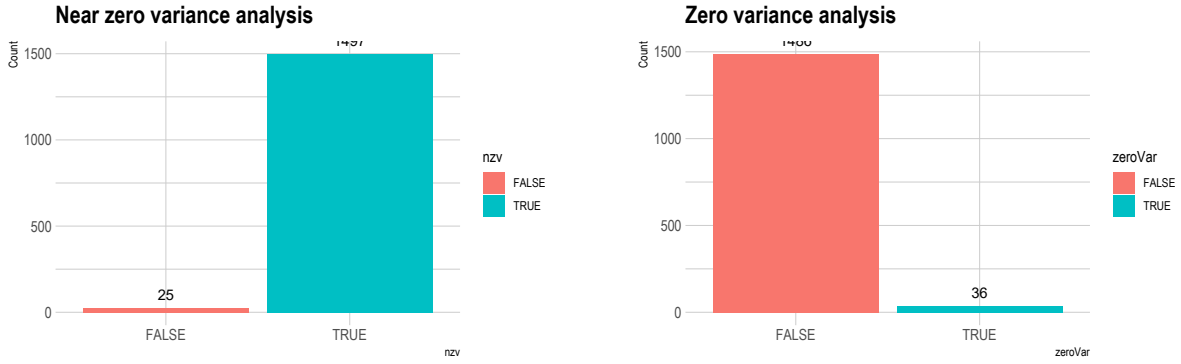
4.0 Analysis

The data set contains 0 NA, with 1763 rows and 1558 features. Each features is a different sensor reading and unfortunately no info was provided about what type of sensor or the unit measure.

As expected the target variable in the data set is very unbalanced. This makes complete sense, since manufacturers have put in place many sophisticated systems to reduce production failures.

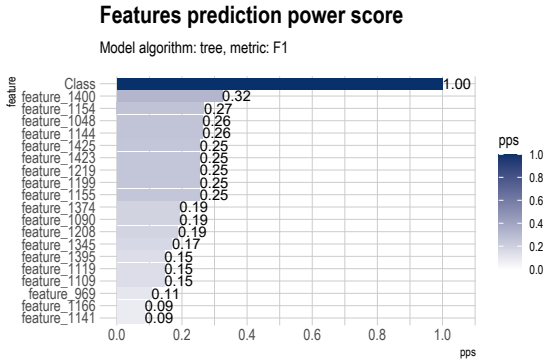


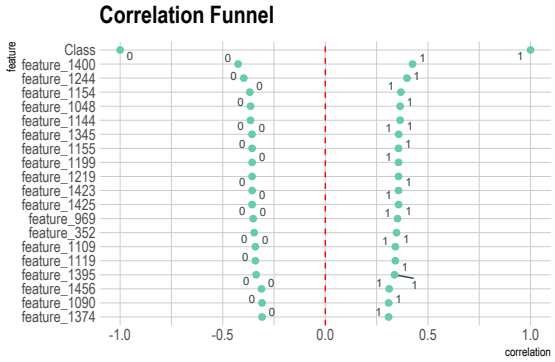
For the same reason we expect also that some sensors has always a constant reading. This means that some features in the data set has zero variance or near zero variance.

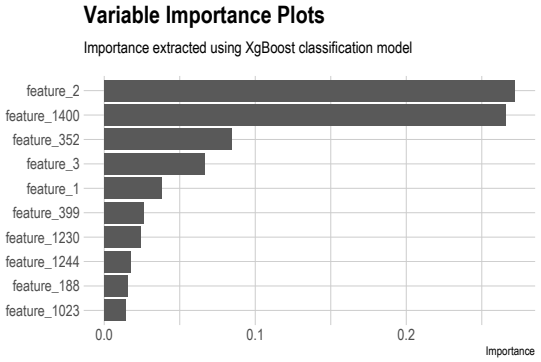


As expected some sensors has near zero or zero variance. Near zero variance features can still have predicting power in this scenario, since even small changes in sensor reading can have a big impact in such complex manufacturing process.

To discover which sensor has the most relevance in the prediction of anomalies prediction power score, correlation funnel and variable importance score were performed.







Several features were acknowledged as very important for the purposes of predicting anomalies in all 3 methods performed. If more info were provided a more depth feature engineering could have been implemented.

5.0 Prediction model

The desired prediction model needs to estimate the probability of each silicon wafer production lot to have anomalies. Taking in consideration the natural unbalanced nature of this data set, using it to directly train the model is strongly not recommended to avoid poor performance on the minority class which in this case is exactly the one needed to be detected.

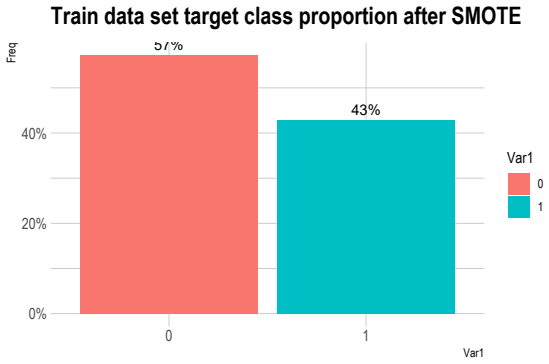
For this reason before train the model a data augmentation technique was applied to the data set partition dedicated to train the model. The technique applied was SMOTE which stands for Synthetic Minority Oversampling Technique and it works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space. For more information, this technique was described by Nitesh Chawla, et al. in their 2002 paper named for the technique titled **"SMOTE: Synthetic Minority Over-sampling Technique."**.

Table 1: Prediction model leaderboard

model_id	auc	logloss	aucpr	mean_per_class_error	rmse	mse
StackedEnsemble_AllModels_AutoML_20210525_173915	0.9795792	0.1669274	0.9474878	0.04083333	0.2043920	0.04177610
StackedEnsemble_BestOfFamily_AutoML_20210525_173915	0.9791167	0.1690365	0.9445563	0.04083333	0.2042882	0.04173369
GLM_1_AutoML_20210525_173915	0.9788375	0.1818361	0.9391775	0.04666667	0.2110859	0.04455726
DRF_1_AutoML_20210525_173915	0.9720125	0.2069787	0.9642818	0.05875000	0.2371691	0.05624917
DeepLearning_1_AutoML_20210525_173915	0.9696000	0.2481286	0.9377460	0.06833333	0.2375114	0.05641166
XRT_1_AutoML_20210525_173915	0.9694042	0.2362604	0.9584812	0.07791667	0.2579024	0.06651364
GBM_4_AutoML_20210525_173915	0.9604083	0.2152173	0.9515836	0.06083333	0.2361188	0.05575209
GBM_2_AutoML_20210525_173915	0.9580583	0.2168621	0.9496298	0.06208333	0.2362886	0.05583230
GBM_3_AutoML_20210525_173915	0.9573333	0.2220667	0.9433997	0.06458333	0.2407075	0.05794008
GBM_1_AutoML_20210525_173915	0.9454792	0.2435467	0.9321256	0.07666667	0.2558990	0.06548431
GBM_grid__1_AutoML_20210525_173915_model_1	0.9448833	0.2894398	0.9345733	0.08208333	0.2827782	0.07996351
GBM_5_AutoML_20210525_173915	0.9424667	0.2779311	0.9360620	0.08000000	0.2761098	0.07623661

After SMOTE was applied the new train data set target class proportion is as follow:



For the prediction model, the approach taken was a generalist one, more a starting point rather than a complex and definitive solution. Using the **H2o.ai** platform in auto machine learning configuration, it was possible to create different models using a variety of machine learning algorithms in the same time.

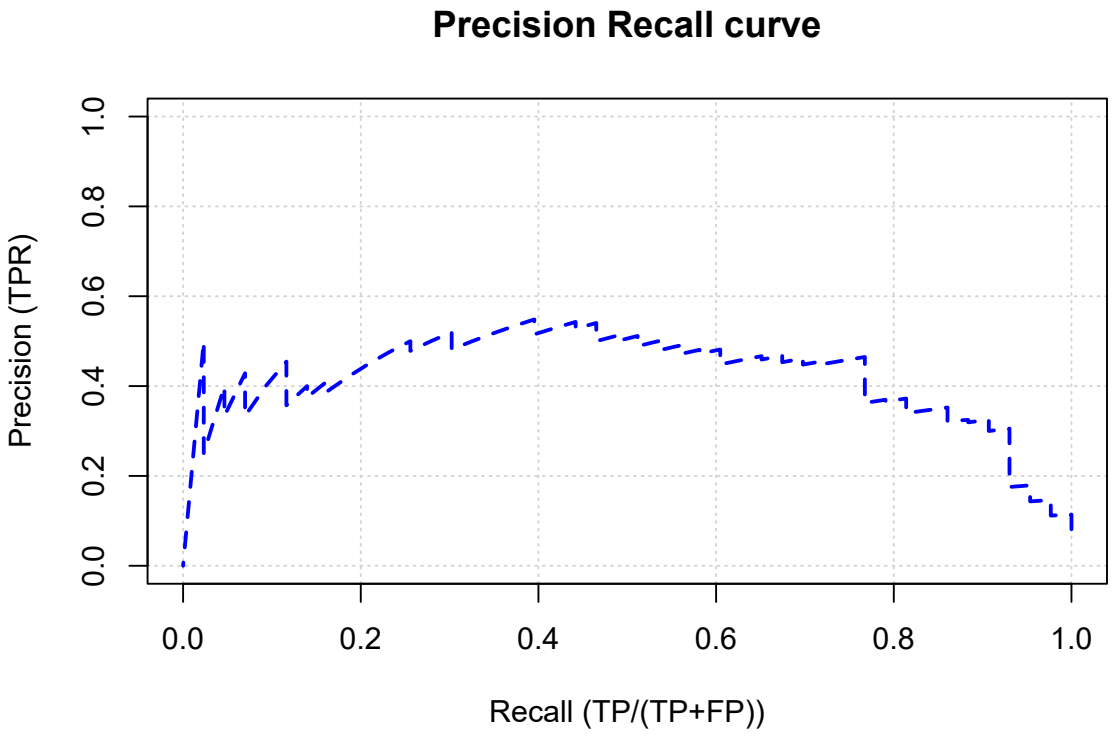
The evaluation of the models was made using AUC metrics.

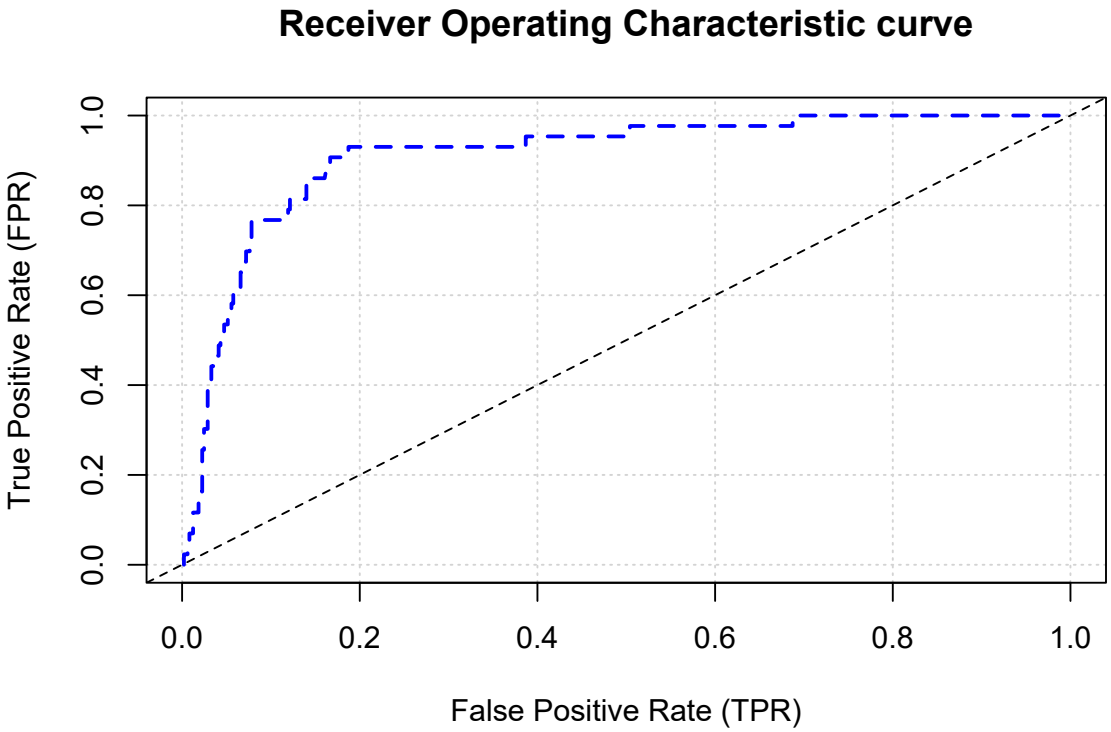
The training process output several model summarized in the following 'leader-board':

The most performing model details are as follow:

```
## H2OBinomialMetrics: stackedensemble
##
## MSE:  0.07393881
## RMSE:  0.2719169
## LogLoss:  0.2737881
## Mean Per-Class Error:  0.1553737
## AUC:  0.910446
## AUCPR:  0.4185134
## Gini:  0.820892
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##          0  1    Error     Rate
## 0      448 38 0.078189  =38/486
## 1       10 33 0.232558   =10/43
## Totals 458 71 0.090737  =48/529
```

```
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold      value idx
## 1                        max f1  0.418513   0.578947  69
## 2                        max f2  0.418513   0.679012  69
## 3                  max f0point5  0.892761   0.523560  35
## 4                  max accuracy  0.939447   0.924386  29
## 5                 max precision  0.939447   0.548387  29
## 6                    max recall  0.009168   1.000000 321
## 7                max specificity  0.996716   0.997942   0
## 8               max absolute_mcc  0.418513   0.552547  69
## 9    max min_per_class_accuracy  0.133854   0.860082 101
## 10 max mean_per_class_accuracy  0.076460   0.871495 126
## 11                      max tns  0.996716 485.000000   0
## 12                      max fns  0.996716  43.000000   0
## 13                      max fps  0.000701 486.000000 399
## 14                      max tps  0.009168  43.000000 321
## 15                      max tnr  0.996716   0.997942   0
## 16                      max fnr  0.996716   1.000000   0
## 17                      max fpr  0.000701   1.000000 399
## 18                      max tpr  0.009168   1.000000 321
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
```





6.0 Conclusion

In this work a classification model for the detection of anomalies in silicon wafer manufacturing, with an acceptable performance, was successfully generated showing that a machine learning approach for this application is feasible and can radically improve the manufacturing process. The performance, even if it is not extreme, is reasonably high considering that the model formulation was unprovided of domain knowledge due to the lack of additional information in the data set.